DDSD – Distributed Database System Design

Wrong Assumptions:

→ DDBS reduced consistency in the absence of partitions is NOT due to CAP based decision making

- C — Consistency
- A — Availability
- P — Partition tolerant

⇒ Doesn't impose restrictions on a DDBS during normal system operation (belief of saying implementing limited system)

⇒ Doesn't constrain any system abilities

→ applys limitations only in the face of certain type of failures

High availability vs High consistency

partition tolerance + Existence of network partition itself (why?)

allows to make complete ACID (with high availability as well if there are no partition)

Data transactional throughput]⇒ elastically scalable DBs

Amazon — Dynamo (core services in e-comm)

Facebook — Cassandra (Inbox search)

LinkedIn — Voldemort (online updates) [write intensive]

Yahoo — PNUTS (user data across sites) (even 100ms)

Latency = Very Critical

Consistency vs latency → more influential than CAP tradeoffs → exists even without network partition

CAP + consistency/latency tradeoffs } ⇒ PACELC

(In the proof of CAP) → Atomic linearizable consistency

PACELC

↳ In case of partition Ⓟ how Availability Ⓐ & consistency Ⓒ tradeoff?

Else Ⓔ how Latency Ⓛ & consistency Ⓒ tradeoff?

Replication of data

because high availability = replication of data

⇓ system running long enough ⇓ atleast few may fail (to prevent unavailability)

⇒ replicate over WAN (highest availability)

Send update to all replicas (at the same time) → passthrough pre-process layer (to maintain consistency) → increased latency; preprocessor itself → multiple machines (or) single machine (geographical latencies) → (replica near, but preprocessor far)

Send to master node first →

- **synchronous** ⇒ wait until all the updates made by master node to replicas
- **asynchronous** ⇓ treats update as if it were completed before complete replication
  - If read across from any node ⇒ inconsistent reads while data propagation
  - If master node overloaded with requests ↳ read requests also will have to wait
  - even if replica close to the read-req initiator ⇓ has to go through master node (which could be far)

Combination of both ⇒ (with synchronous read and async write)

If R + W > N in a quorum protocol → atleast one node has both the operation } ⇒ consistency = preserved

⇓ If R + W ≤ N ⇒ inconsistent reads

⇓ Quorum protocol not sufficient to guarantee consistency